My experience about memory limitation in embedded system

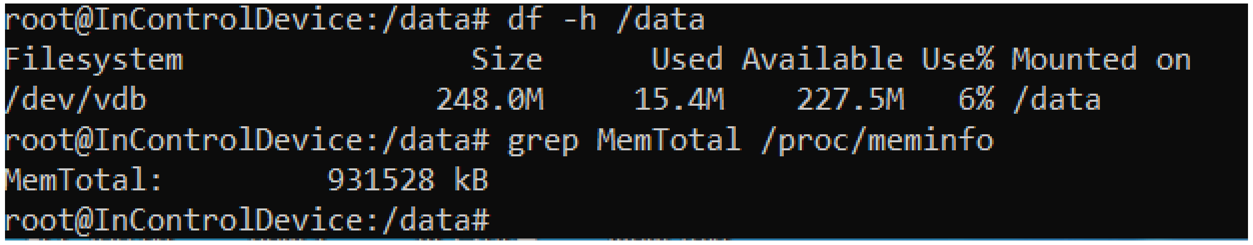
Contents

VC Development Center Vietnam

Hanoi, October 2020

Overview

- 1, Maximum size of disk part and RAM that HBD works on



```
root@InControlDevice:/data# df -h /data
Filesystem                Size      Used Available Use% Mounted on
/dev/vdb                248.0M     15.4M    227.5M   6% /data
root@InControlDevice:/data# grep MemTotal /proc/meminfo
MemTotal:         931528 kB
root@InControlDevice:/data#
```

  - HBD uses max 10MB/248MB in disk

- Purpose:

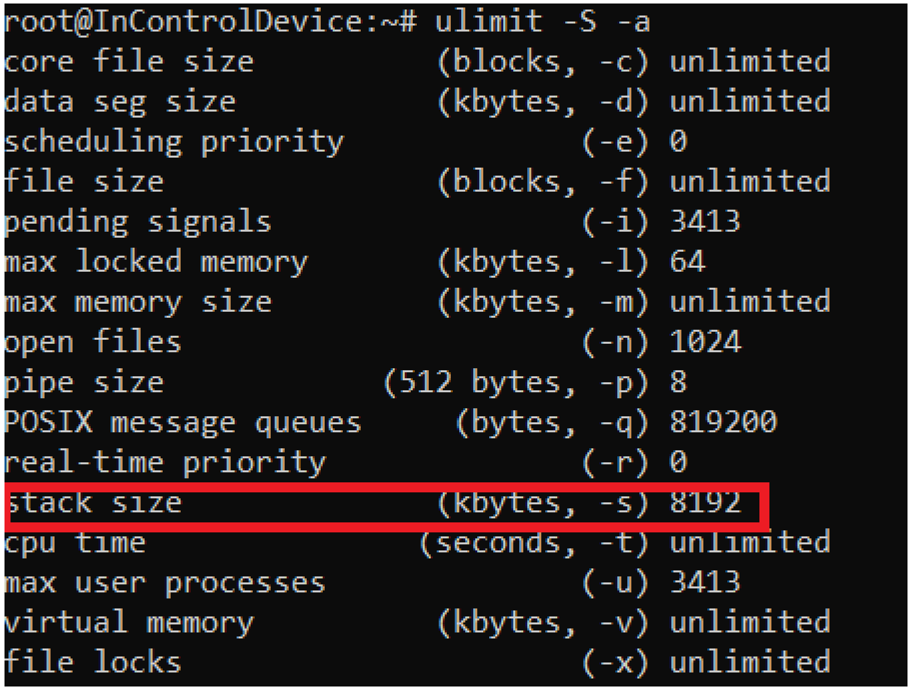
- Even though the data size of HBD is small but in the combination of many applications and services some cases can be occurred.

1, Full disk -> application crashes

2, Full RAM -> application crashes

Allocate large size memory in RAM

- Allocating large size memory on stack is not available since the OS limited stack size = 8MB



```
root@InControlDevice:~# ulimit -S -a
core file size          (blocks, -c) unlimited
data seg size           (kbytes, -d) unlimited
scheduling priority             (-e) 0
file size               (blocks, -f) unlimited
pending signals                 (-i) 3413
max locked memory       (kbytes, -l) 64
max memory size         (kbytes, -m) unlimited
open files                      (-n) 1024
pipe size            (512 bytes, -p) 8
POSIX message queues     (bytes, -q) 819200
real-time priority              (-r) 0
stack size              (kbytes, -s) 8192
cpu time               (seconds, -t) unlimited
max user processes              (-u) 3413
virtual memory          (kbytes, -v) unlimited
file locks                      (-x) unlimited
```

Allocate large size memory in RAM

- **To avoid application crashes if RAM is full in dynamic memory allocation, one normal method is catching “std::bad_alloc” to stop operation**

```
struct Data
{
    char value[1024];
};

int main()
{
    // Max size RAM 32 bits = 4GB = 4 * 1024 * 1024 * 1024
    std::vector<Data> buffer;
    try
    {
        buffer.reserve(2 * 1024 * 1024);
    }
    catch (std::bad_alloc & e)
    {
        std::cout << e.what() << std::endl;
        std::cout << "Do nothing" << std::endl;
    }
    std::cout << "buffer size = " << buffer.size() << std::endl; // buffer size = 0
    return 0;
}
```

Allocate large size memory in RAM

- **But in some cases that are suitable with HBD we can resize buffer size < max size to continue working with this buffer**.

```
int main()
{
    // Max size RAM 32 bits = 4GB = 4 * 1024 * 1024 * 1024
    std::vector<std::shared_ptr<Data>> buffer;
    try
    {
        constexpr size_t size = 2 * 1024 * 1024;
        buffer.reserve(size);
        for (size_t i = 0; i < size; i++)
        {
            buffer.push_back(std::make_shared<Data>());
        }
    }
    catch (std::bad_alloc& e)
    {
        std::cout << e.what() << std::endl;
        buffer.resize(buffer.size() > 0? buffer.size() -1 : 0); // resize buffer to have more memory for other actions
    }
    std::cout << "buffer size = " << buffer.size()*sizeof(Data) << std::endl; // buffer size > 0
    return 0;
}
```

Limit data size in disk

- **Ring buffer basics**:

https://www.embedded.com/ring-buffer-basics/

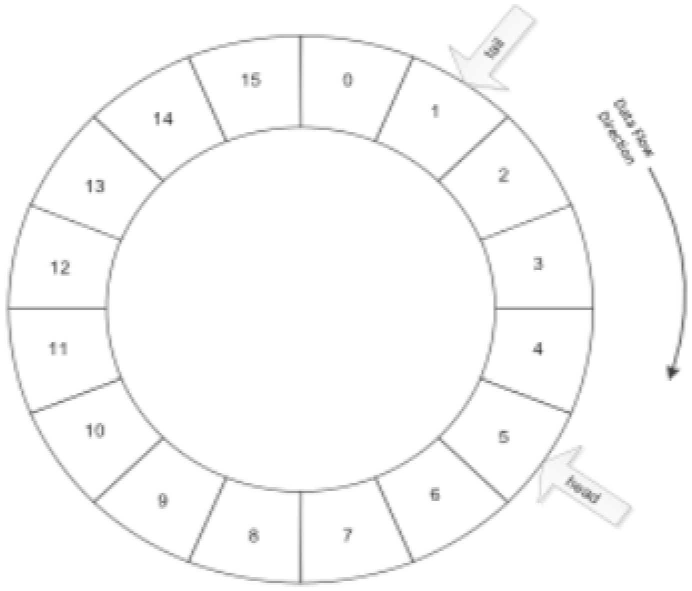


Figure 1: Structure of a ring buffer.

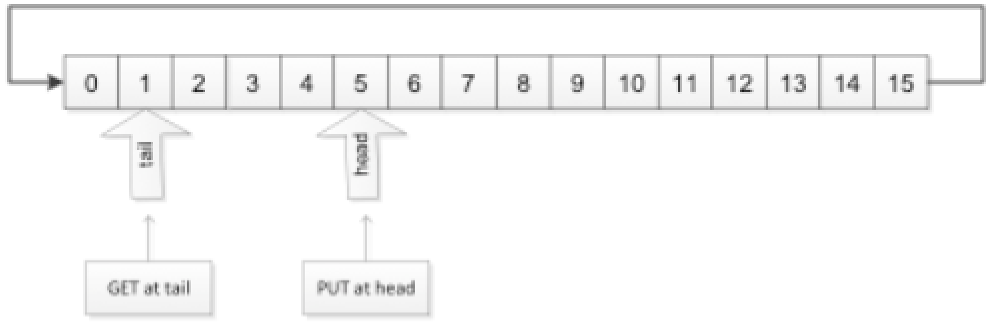


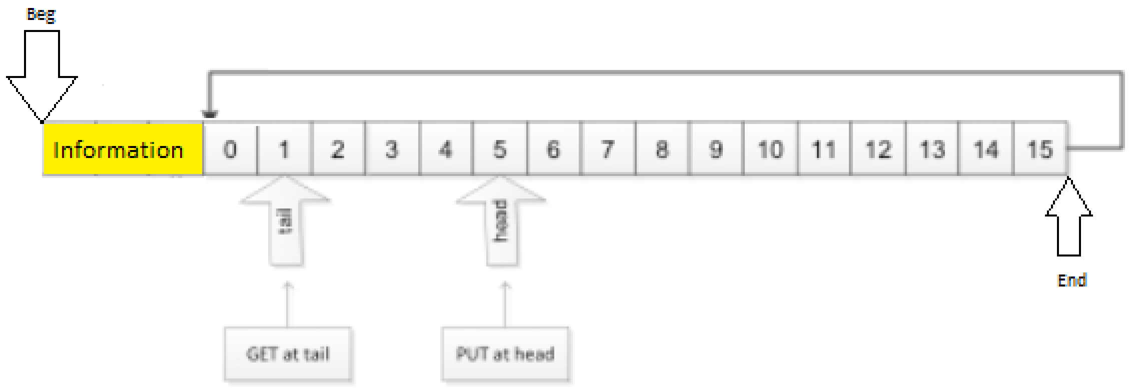
Figure 2: Linear buffer implementation of the ring buffer.

Information needed: head index , tail index , size

Limit data size in disk

- **Implement ring buffer:**

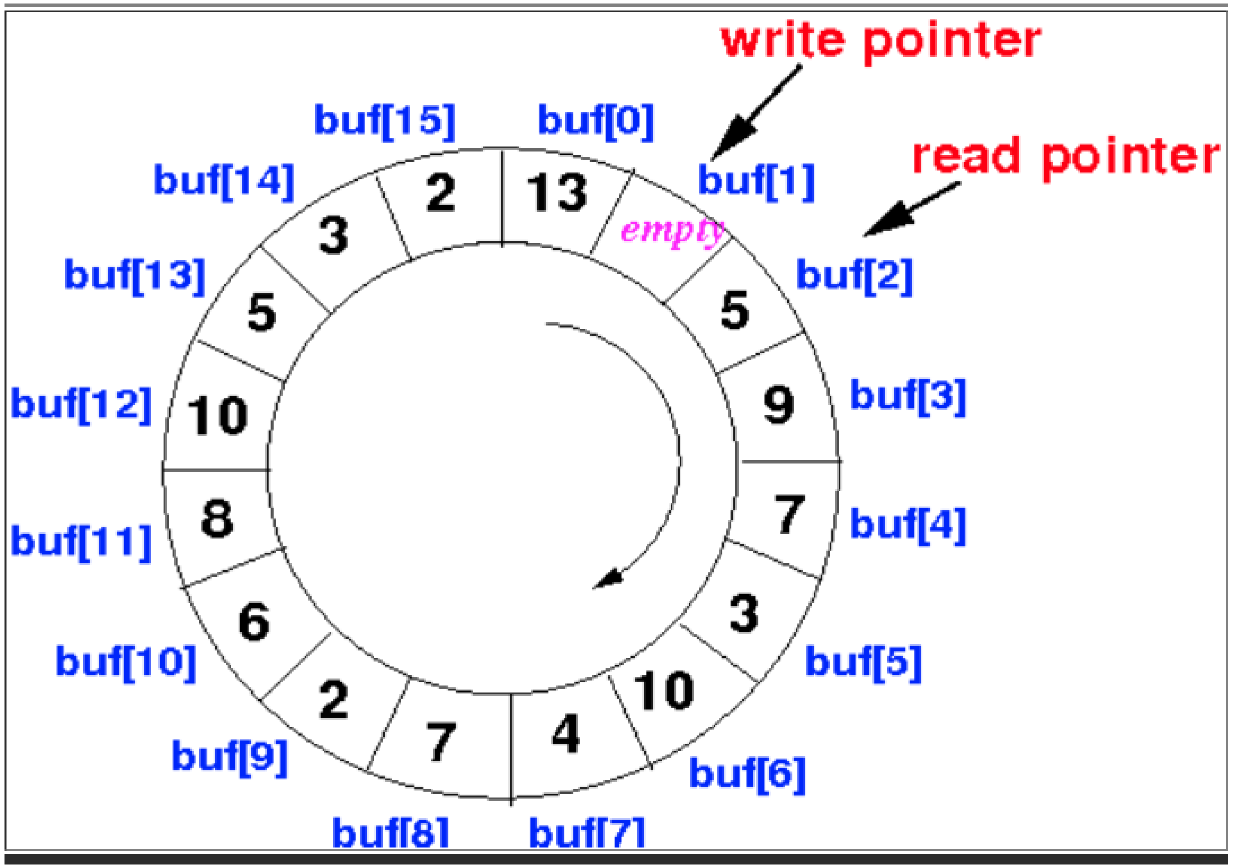
```
template <class Data>
class InternalDataStorage
{
    struct Information
    {
        uint32_t head = 0;
        uint32_t tail = 0;
    } information;

    std::string path;
    std::ofstream dataSaved;
    std::ifstream dataRead;
    size_t size = 0;
public:
    InternalDataStorage(const std::string& path, size_t size);
    bool push(const Data& data);
    bool pop();
    std::vector<std::shared_ptr<Data>> readBlock(size_t blockSize);
    void deleteBlock(size_t blockSize);
private:
    bool saveInformation();
    bool openFileToSave();
    bool openFileToRead();
    inline uint32_t getOffset() { return sizeof(Information); }
};
```



Limit data size in disk

- **Ring buffer basics**:

One must be able to handle the case where the queue is full



Limit data size in disk

- **Implement ring buffer:**

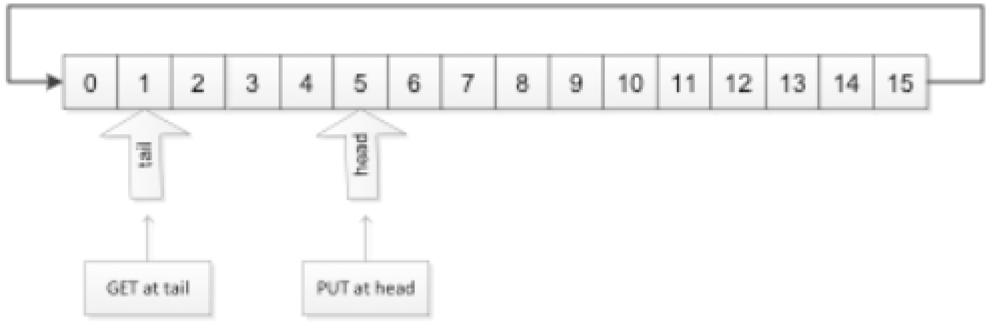


Figure 2: Linear buffer implementation of the ring buffer.

```
template<class Data>
bool InternalDataStorage<Data>::push(const Data& data)
{
    //{
    // TODO: write data to file
    //}
    if (++information.head >= size) information.head = 0;
    if (information.head == information.tail)
    {
        if (++information.tail >= size) information.tail = 0;
    }
    return saveInformation();//save head-tail to file
}
```

```
template<class Data>
bool InternalDataStorage<Data>::pop()
{
    if (information.head == information.tail)
    {
        return true;
    }
    if (++information.tail >= size) information.tail = 0;
    return saveInformation();//save head-tail to file
}
```

Thank you